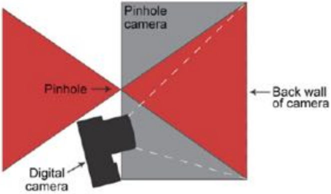
Cameras and Images

Pinhole Camera

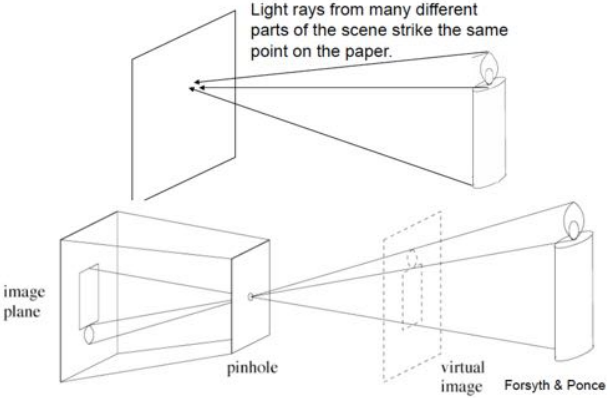
[Source: A. Torralba]



- Make your own camera
- http://www.foundphotography.com/PhotoThoughts/archives/2005/04/pinhole_camera_2.html

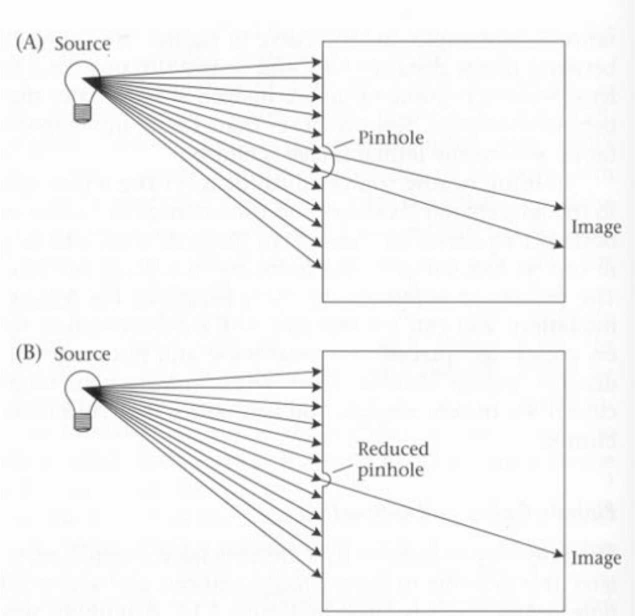
Pinhole Camera – How It Works

[Source: A. Torralba]



- The pinhole camera only allows rays from one point in the scene to strike each point of the paper.

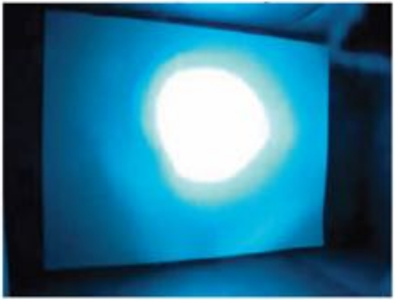
Pinhole Camera – How It Works



ɔurce: A. Torralba]

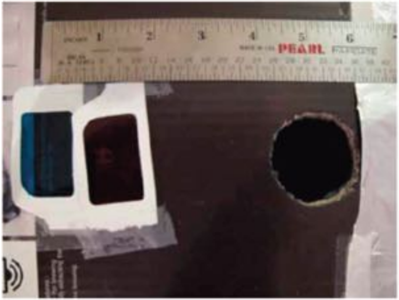
Pinhole Camera – Example

[Source: A. Torralba]



Pinhole Camera

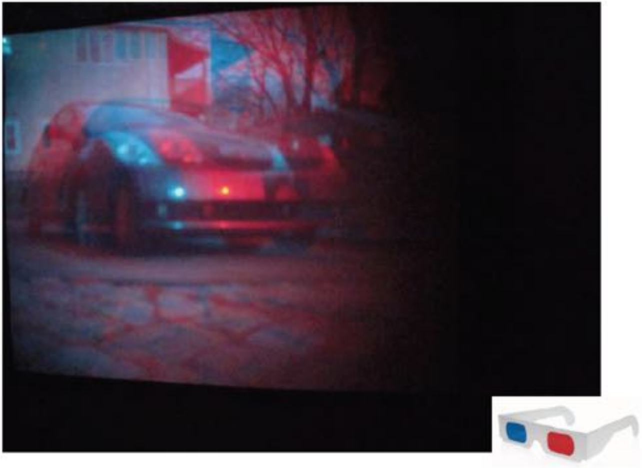
[Source: A. Torralba]



- You can make it stereo

Pinhole Camera – Stereo Example

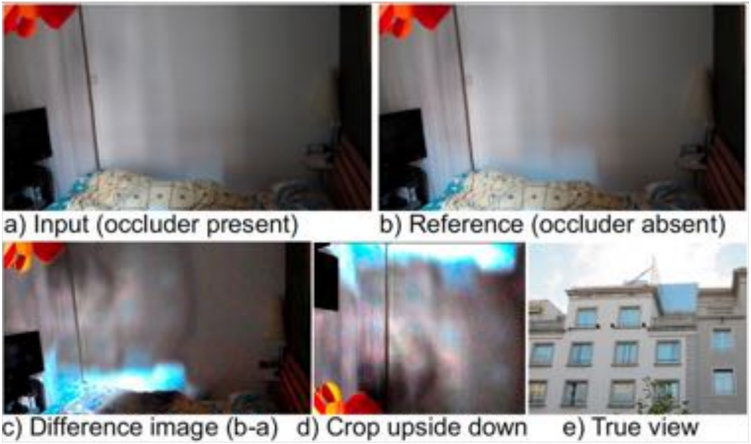
[Source: A. Torralba]



- Try it with 3D glasses!

Pinhole Camera

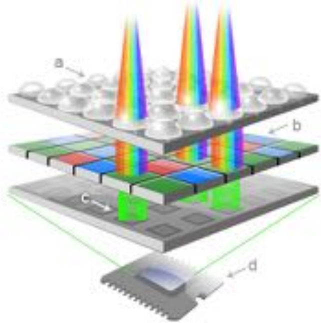
[Source: A. Torralba]



- Remember this example?
- In this case the window acts as a pinhole camera into the room

Digital Camera

[Adopted from S. Seitz]



- A digital camera replaces film with a sensor array
- Each cell in the array is a light-sensitive diode that converts photons to electrons
- http://electronics.howstuffworks.com/cameras-photography/digital/digital-camera.htm

Image Formation

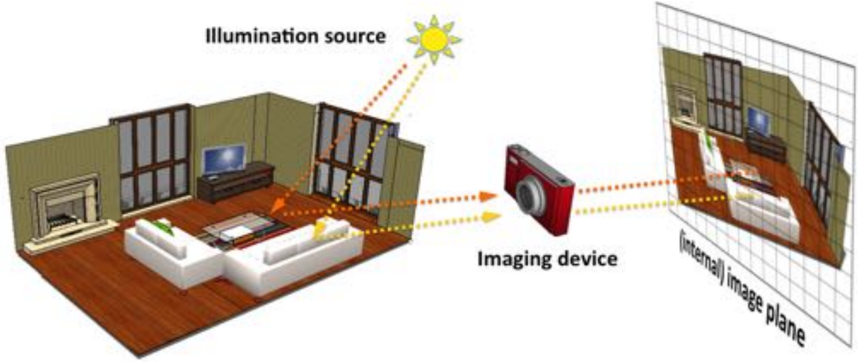
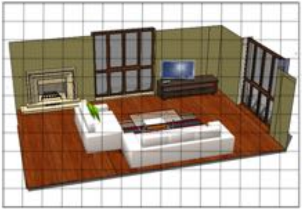


Image formation process producing a particular image depends on:

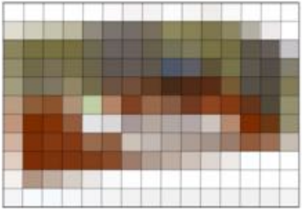
- lighting conditions
- scene geometry
- surface properties
- camera optics

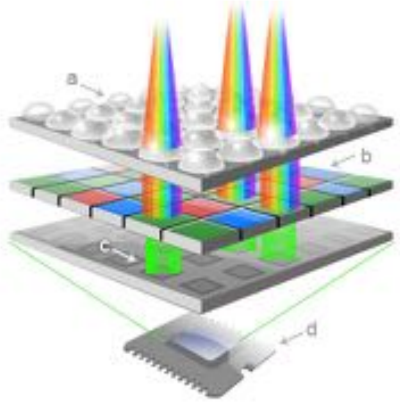
Digital Image

Continuous image projected to sensor array



Sampling and quantization



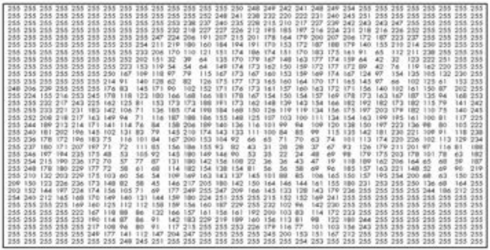


http://pho.to/media/images/digital/digital-sensors.jpg

- Sample the 2D space on a regular grid
- Quantize each sample (round to nearest integer)

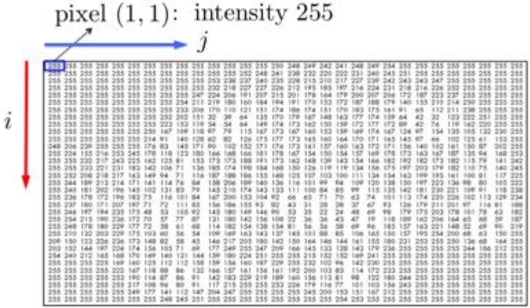
Digital Image

- Image is a matrix with integer values
- We will typically denote it with I



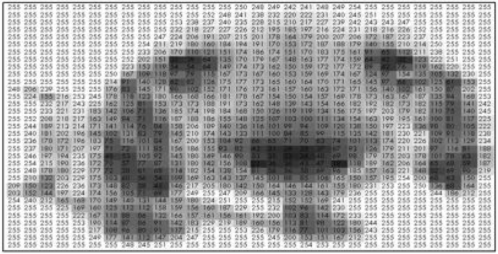
Digital Image

- Image is a matrix with integer values
- We will typically denote it with I
- $I(i,j)$ is called **intensity**



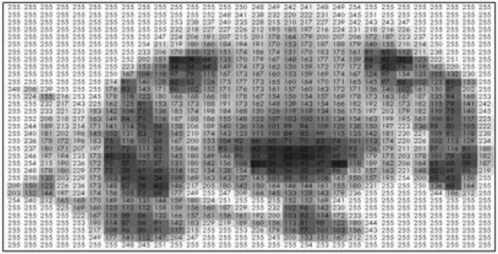
Digital Image

- Image is a matrix with integer values
- We will typically denote it with I
- $I(i, j)$ is called **intensity**
- Matrix I can be $m \times n$ (grayscale)



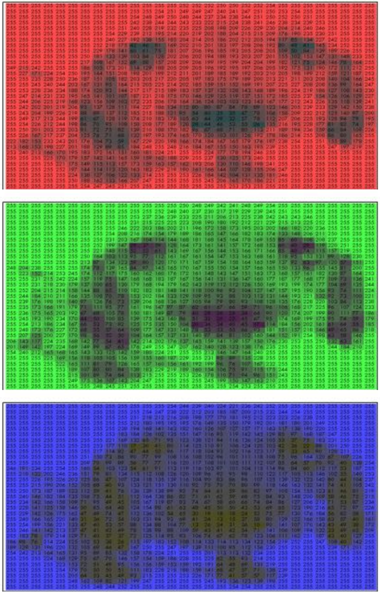
Digital Image

- Image is a matrix with integer values
- We will typically denote it with I
- $I(i, j)$ is called **intensity**
- Matrix I can be $m \times n$ (grayscale)
- or $m \times n \times 3$ (color)

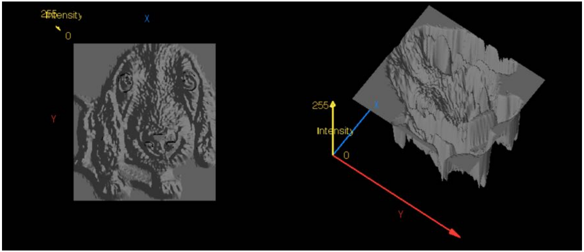


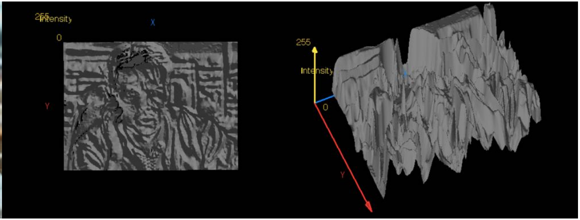
Digital Image

- Image is a matrix with integer values
- We will typically denote it with I
- $I(i,j)$ is called **intensity**
- Matrix I can be $m \times n$ (grayscale)
- or $m \times n \times 3$ (color)



Intensity

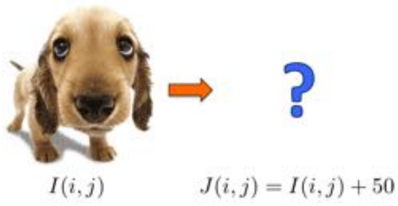




- We can think of a (grayscale) image as a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ giving the intensity at position (i, j)
- Intensity 0 is black and 255 is white

Image Transformations

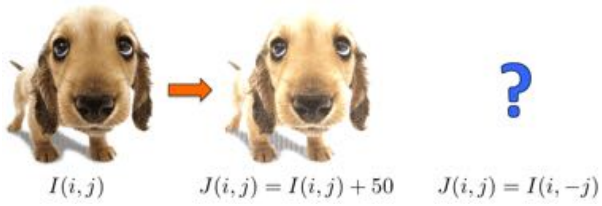
- As with any function, we can apply operators to an image, e.g.:



- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

- As with any function, we can apply operators to an image, e.g.:

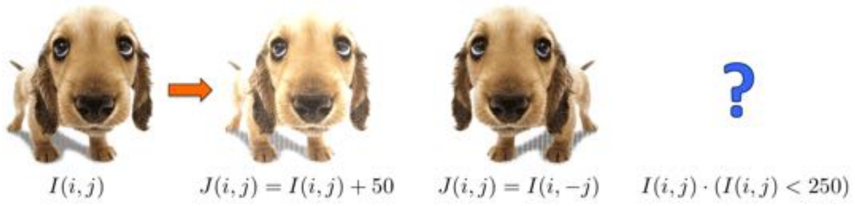


- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

Image Transformations

- As with any function, we can apply operators to an image, e.g.:

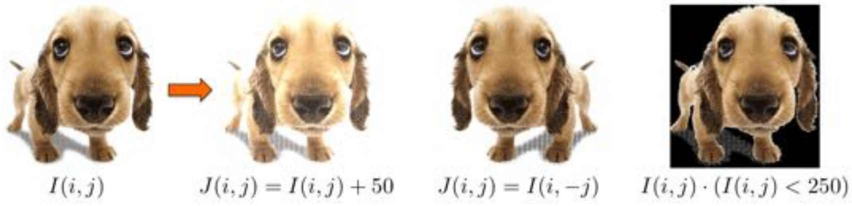


- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

Image Transformations

- As with any function, we can apply operators to an image, e.g.:



- We'll talk about special kinds of operators, **correlation** and **convolution** (linear filtering)

[Adapted from: N. Snavely]

Linear Filters

Reading: Szeliski book, Chapter 3.2

Motivation: Finding Waldo

- How can we find Waldo?

[Source: R. Urtasun]

Answer

- Slide and compare!
- In formal language: **filtering**

Motivation: Noise reduction

- Given a camera and a still scene, how can you reduce noise?

[Source: S. Seitz]

Image Filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel
- In other words... Filtering

10	5	3
4	5	1
1	1	7

Local image data

Some function

	7	

Modified image data

[Source: L. Zhang]

Applications of Filtering

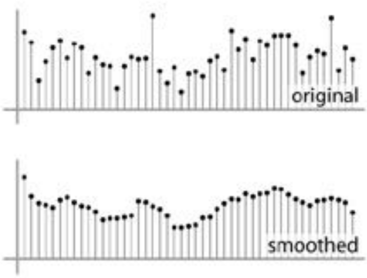
- Enhance an image, e.g., **denoise**.
- Detect patterns, e.g., **template matching**.
- Extract information, e.g., **texture, edges**.
- Filtering is used in Convolutional Neural Networks

Applications of Filtering

- Enhance an image, e.g., **denoise**. Let's talk about this first
- Detect patterns, e.g., **template matching**.
- Extract information, e.g., **texture, edges**.

Noise reduction

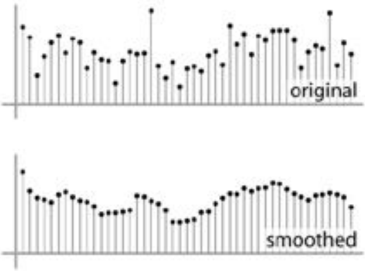
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

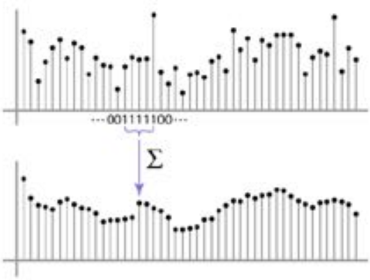
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.



[Source: S. Marschner]

Noise reduction

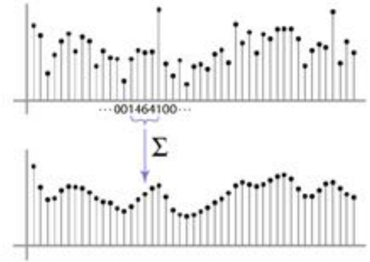
- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- **Moving average** in 1D: [1, 1, 1, 1, 1]/5



[Source: S. Marschner]

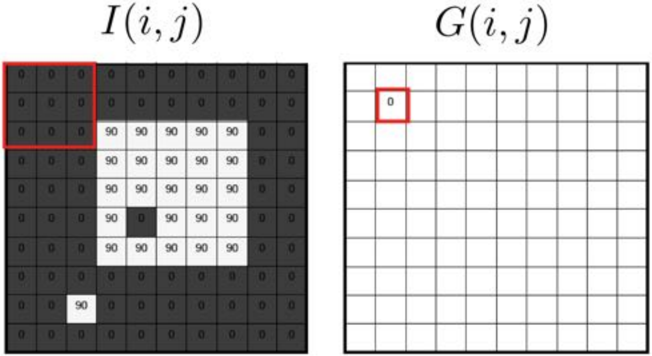
Noise reduction

- Simplest thing: replace each pixel by the average of its neighbors
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Non-uniform weights [1, 4, 6, 4, 1] / 16



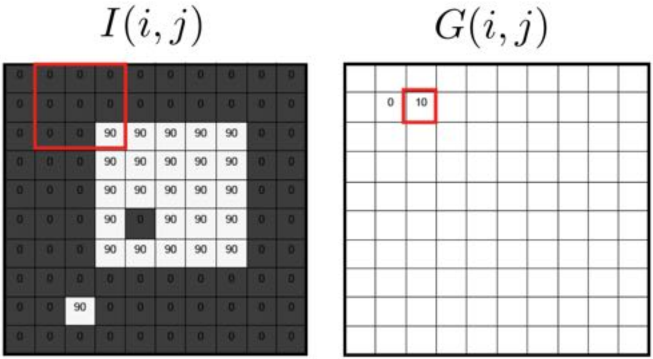
[Source: S. Marschner]

Moving Average in 2D



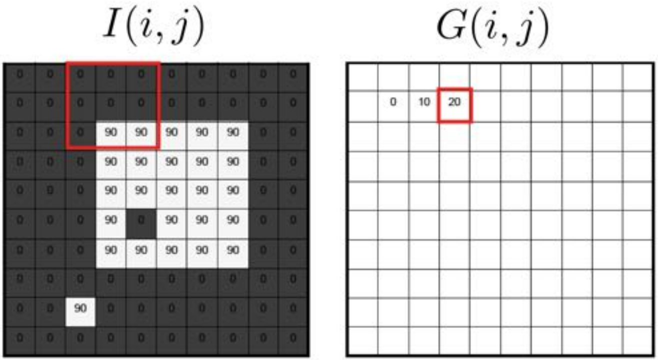
[Source: S. Seitz]

Moving Average in 2D



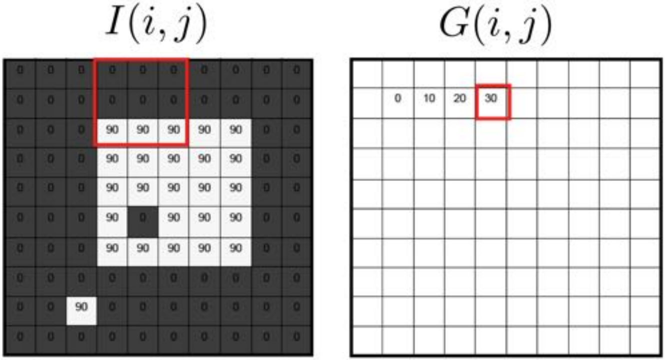
[Source: S. Seitz]

Moving Average in 2D



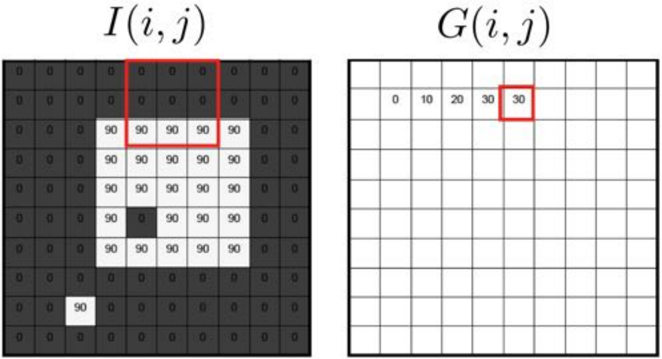
[Source: S. Seitz]

Moving Average in 2D



[Source: S. Seitz]

Moving Average in 2D



[Source: S. Seitz]

Moving Average in 2D

$I(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i,j)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

[Source: S. Seitz]

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- The entries of the weight **kernel** or **mask** $F(u,v)$ are often called the **filter coefficients**.

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- The entries of the weight **kernel** or **mask** $F(u,v)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$G = F \otimes I$$

Linear Filtering: Correlation

- Involves weighted combinations of pixels in small neighborhoods:

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

- The output pixels value is determined as a weighted sum of input pixel values

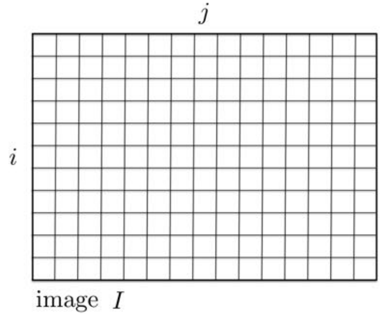
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

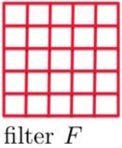
- The entries of the weight **kernel** or **mask** $F(u,v)$ are often called the **filter coefficients**.
- This operator is the **correlation** operator

$$G = F \otimes I$$

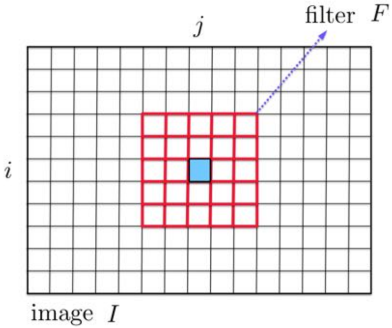
Linear Filtering: Correlation

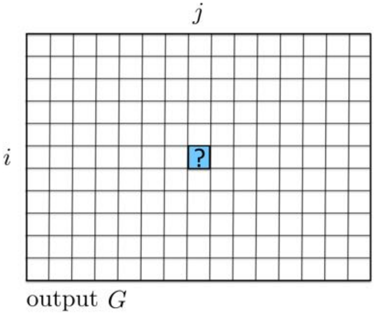
- It's really easy!



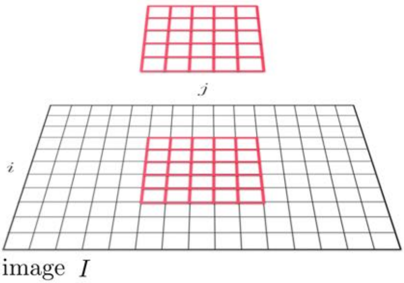


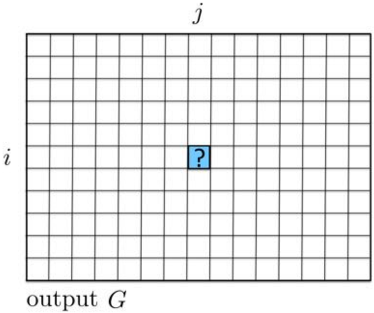
- It's really easy!



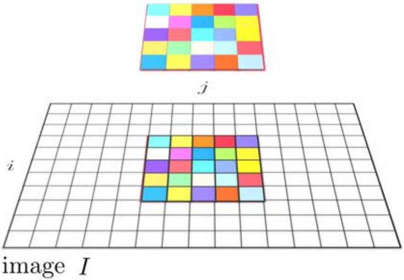


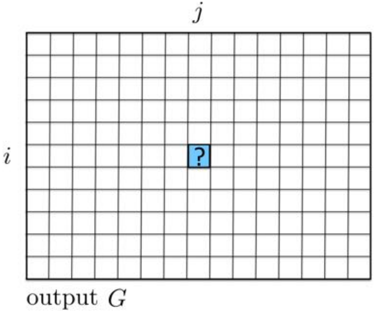
- It's really easy!



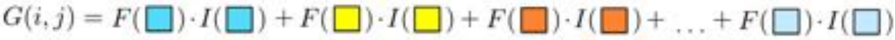


- It's really easy!



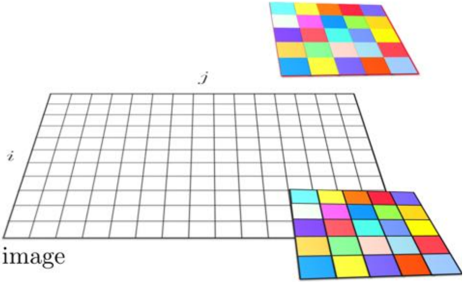


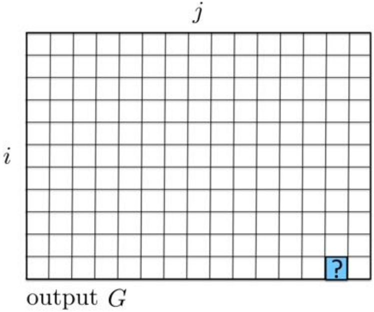
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



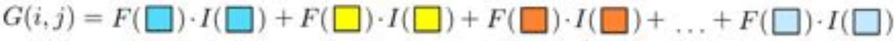
Linear Filtering: Correlation

- What happens along the borders of the image?





$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



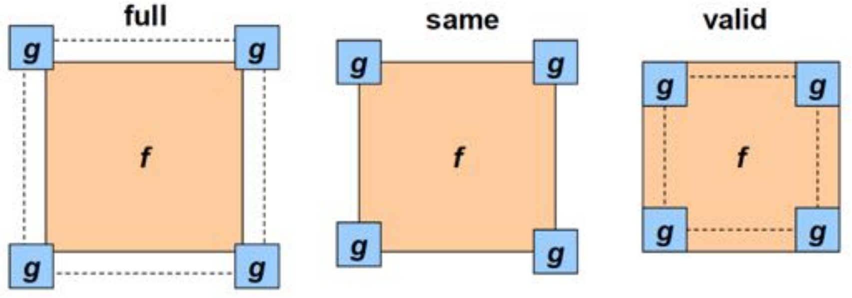
Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: FILTER2(G, F, SHAPE)
  Python: SCIPY.NDIMAGE.CONVOLVE
- shape = "full" output size is sum of sizes of f and g
- shape = "same": output size is same as f
- shape = "valid": output size is difference of sizes of f and g

[Source: S. Lazebnik]

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
- MATLAB: FILTER2(G, F, SHAPE)
  Python: SCIPY.NDIMAGE.CONVOLVE
- shape = "full" output size is sum of sizes of f and g
- shape = "same": output size is same as f
- shape = "valid": output size is difference of sizes of f and g



Filtering with Correlation: Example

- What's the result?

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

?

[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?

Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

Filtered
(no change)

[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?

Original

0	0	0
0	0	1
0	0	0

?

[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?

0	0	0
0	0	1
0	0	0

[Source: D. Lowe]

- What's the result?

Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

[Source: D. Lowe]

Filtering with Correlation: Example

- What's the result?

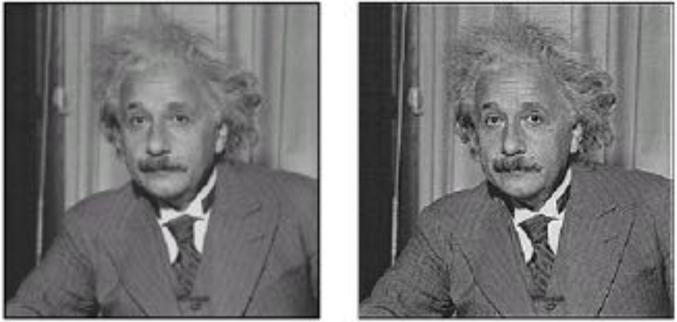
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

Sharpening filter
(accentuates edges)

[Source: D. Lowe]

Sharpening



before after

[Source: D. Lowe]

Sharpening

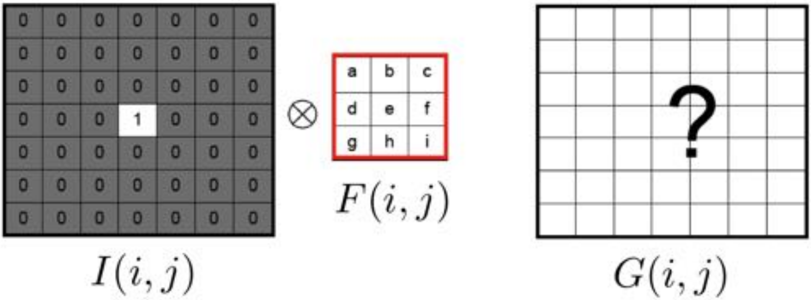
unfiltered

filtered

[Source: N. Snavely]

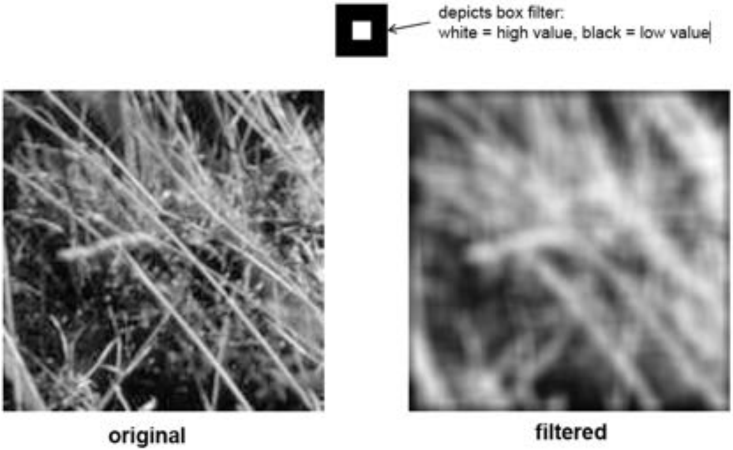
Example of Correlation

- What is the result of filtering the impulse signal (image) I with the arbitrary filter F?



[Source: K. Grauman]

Smoothing by averaging



- What if the filter size was 5 x 5 instead of 3 x 3?

[Source: K. Graumann]

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?
- Removes high-frequency components from the image (low-pass filter).

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

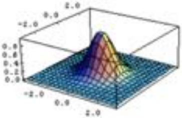
$I(i,j)$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$F(i,j)$

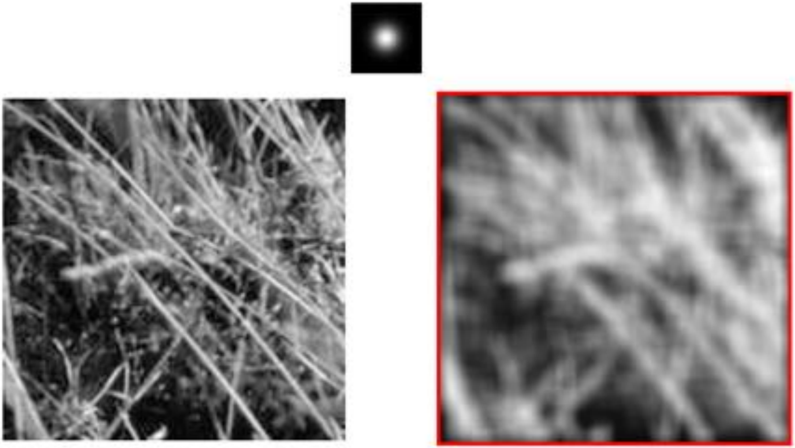
This kernel is an approximation of a 2d Gaussian function:

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



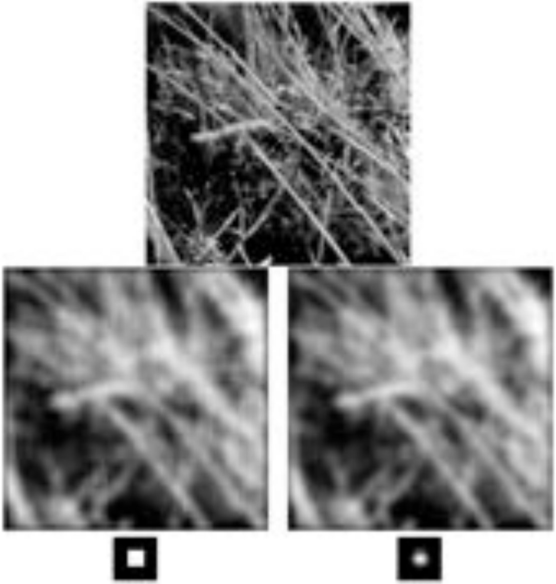
[Source: S. Seitz]

Smoothing with a Gaussian



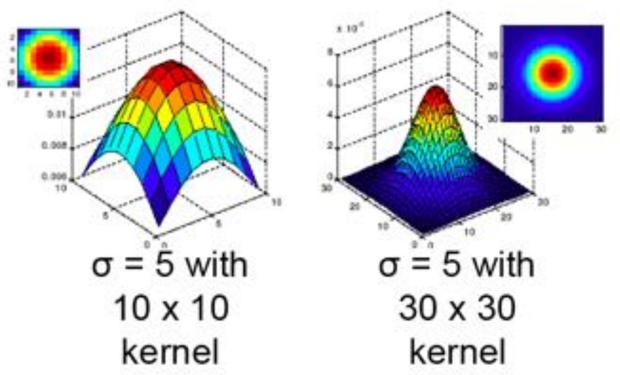
[Source: K. Grauman]

Mean vs Gaussian



Gaussian filter: Parameters

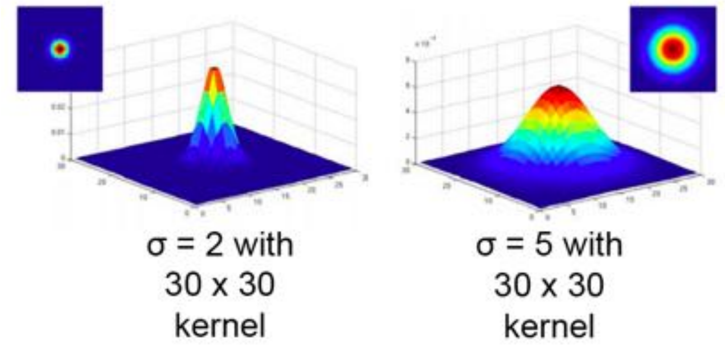
- **Size of filter or mask**: Gaussian function has infinite support, but discrete filters use finite kernels.



[Source: K. Grauman]

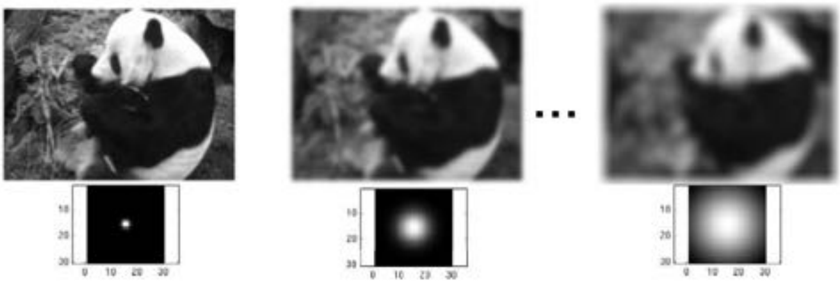
Gaussian filter: Parameters

- **Variance of the Gaussian**: determines extent of smoothing.



[Source: K. Grauman]

Gaussian filter: Parameters



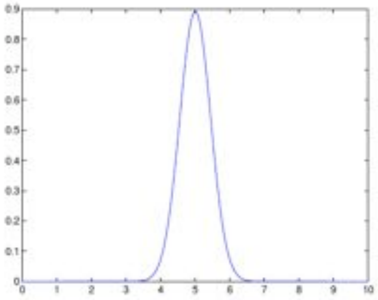
```
for sigma=1:3:10
   h = fspecial('gaussian', fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

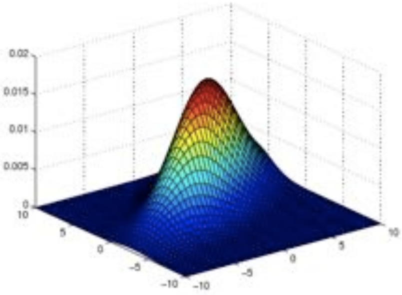
[Source: K. Grauman]

Is this the most general Gaussian?

- No, the most general form for $\mathbf{x} \in \Re^d$

$$\mathcal{N}(\mathbf{x};\, \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$





- We typically use isotropic filters (i.e., circularly symmetric)

Properties of the Smoothing

- All values are positive.
- They all sum to 1.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.
- Remove high-frequency components; low-pass filter.

Note: This holds for smoothing filters, not general filters

Properties of the Smoothing

- All values are positive.
- They all sum to 1.
- Amount of smoothing proportional to mask size.
- Remove high-frequency components; low-pass filter.

Note: This holds for smoothing filters, not general filters

Finding Waldo

image I

- How can we use what we just learned about filtering to find Waldo?

Finding Waldo

image I

filter F

- Is correlation a good choice?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

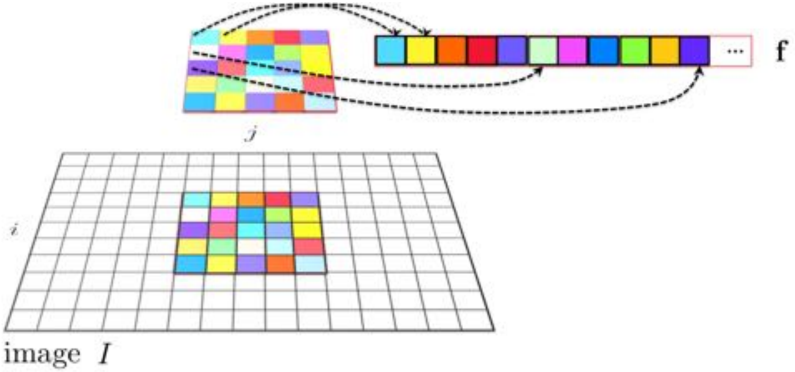
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- Can we write that in a more compact form (with vectors)?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

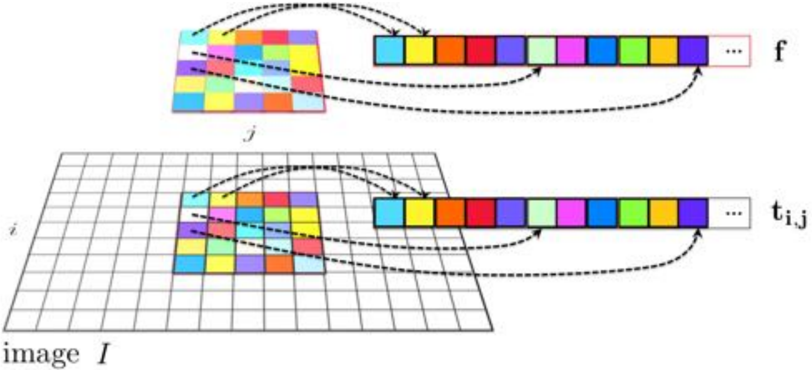
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



A Slight Detour: Correlation in Matrix Form

- Remember correlation:

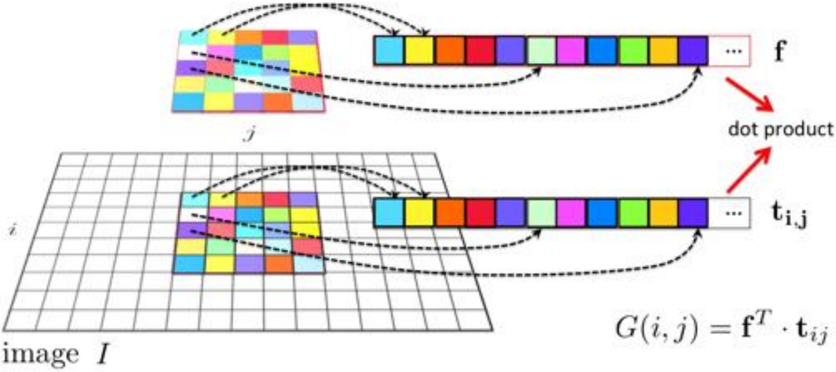
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i-k : i+k, j-k : j+k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

where $\cdot$ is a dot product

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i-k:i+k, j-k:j+k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

  where $\cdot$ is a dot product
- **Homework:** Can we write full correlation $G = F \otimes I$ in matrix form?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i-k : i+k, j-k : j+k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

  where $\cdot$ is a dot product
- Finding Waldo: How could we ensure to get the best "score" (e.g. 1) for an image crop that looks exactly like our filter?

A Slight Detour: Correlation in Matrix Form

- Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

- Can we write that in a more compact form (with vectors)?
- Define $\mathbf{f} = F(:)$, $T_{ij} = I(i-k:i+k, j-k:j+k)$, and $\mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f}^T \cdot \mathbf{t}_{ij}$$

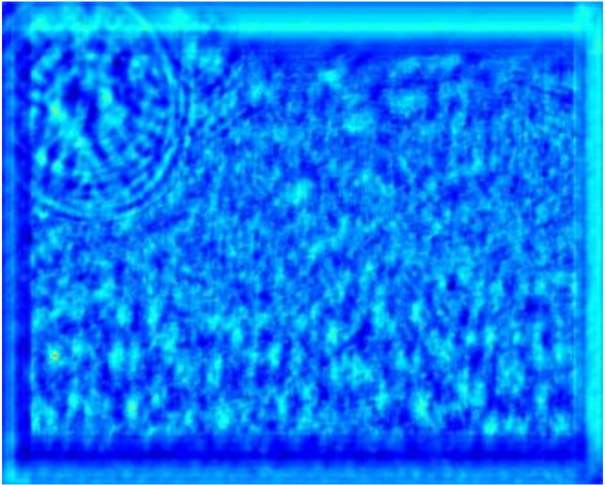
  where $\cdot$ is a dot product
- Finding Waldo: How could we ensure to get the best "score" (e.g. 1) for an image crop that looks exactly like our filter?
- **Normalized cross-correlation:**

$$G(i,j) = \frac{\mathbf{f}^T \cdot \mathbf{t}_{ij}}{||\mathbf{f}|| \cdot ||\mathbf{t_{ij}}||}$$

image I

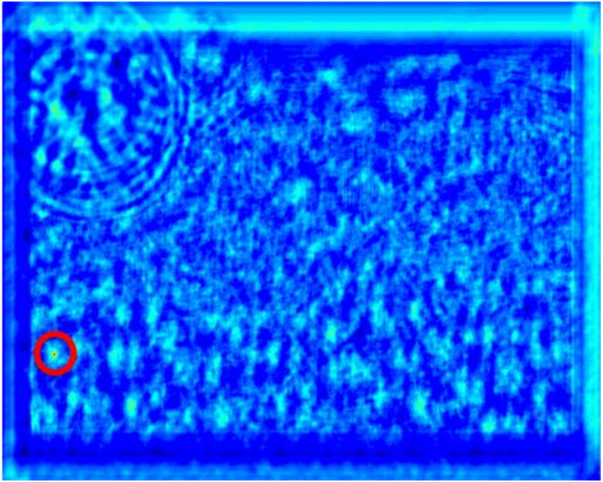
filter F

Back to Waldo



- Result of normalized cross-correlation

Back to Waldo



- Find the highest peak

Back to Waldo

And put a bounding box (rectangle the size of the template) at the point!

Back to Waldo

- **Homework**: Do it yourself! Code on class webpage. Don't cheat!

Convolution

- **Convolution** operator

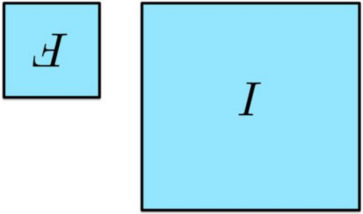
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u, j-v)$$

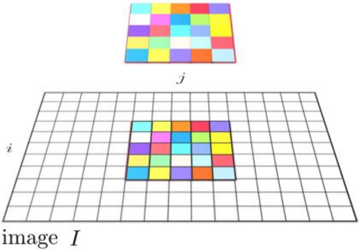
Convolution

- **Convolution** operator

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u, j-v)$$

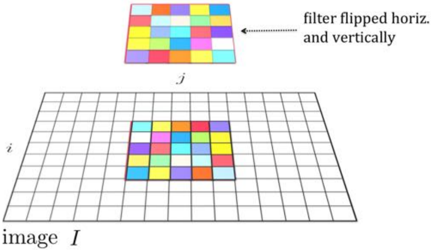
- **Equivalent** to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.





Correlation

=



Convolution

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?
- How will the outputs differ for:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Correlation vs Convolution

- For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?
- How will the outputs differ for:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

- If the input is an impulse signal, how will the outputs differ? $\delta * I$ and $\delta \otimes I$?

- Camera Shake

=

*

Figure: Fergus, et al., SIGGRAPH 2006

- Blur in out-of-focus regions of an image.

Figure: Bokeh: `http://lullaby.homepage.dk/diy-camera/bokeh.html`
Click for more info

[Source: N. Snavely]

Properties of Convolution

$$\begin{aligned}
\text{Commutative} &: f * g = g * f \\
\text{Associative} &: f * (g * h) = (f * g) * h \\
\text{Distributive} &: f * (g + h) = f * g + f * h \\
\text{Assoc. with scalar multiplier} &: \lambda \cdot (f * g) = (\lambda \cdot f) * h
\end{aligned}$$

Properties of Convolution

$$\begin{aligned} \text{Commutative} &: f * g = g * f \\ \text{Associative} &: f * (g * h) = (f * g) * h \\ \text{Distributive} &: f * (g + h) = f * g + f * h \\ \text{Assoc. with scalar multiplier} &: \lambda \cdot (f * g) = (\lambda \cdot f) * h \end{aligned}$$

- The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

Properties of Convolution

$$\begin{aligned} \text{Commutative} &: f * g = g * f \\ \text{Associative} &: f * (g * h) = (f * g) * h \\ \text{Distributive} &: f * (g + h) = f * g + f * h \\ \text{Assoc. with scalar multiplier} &: \lambda \cdot (f * g) = (\lambda \cdot f) * h \end{aligned}$$

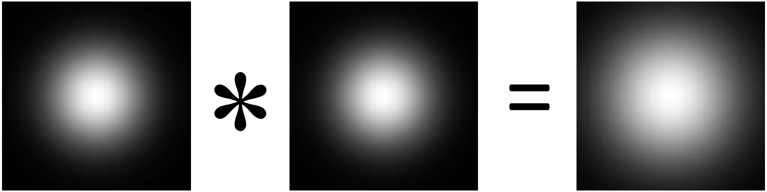
- The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

- **Homework:** Why is this good news?
- **Hint:** Think of complexity of convolution and Fourier Transform
- Both correlation and convolution are **linear shift-invariant (LSI) operators**: the effect of the operator is the same everywhere.

Gaussian Filter

- Convolving twice with Gaussian kernel of width σ is the same as convolving once with kernel of width $\sigma\sqrt{2}$



- We don't need to filter twice, just once with a bigger kernel

[Source: K. Grauman]

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only** $2K$ **operations**.

Separable Filters: Speed-up Trick!

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only** $2K$ **operations**.
- If this is possible, then the convolution filter is called **separable**.

Separable Filters: Speed-up Trick!

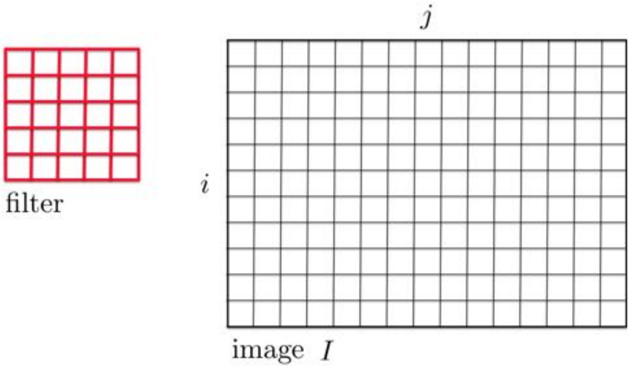
- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter.
- Can we do faster?
- In many cases (**not all!**), this operation can be speed up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only** $2K$ **operations**.
- If this is possible, then the convolution filter is called **separable**.
- And it is the outer product of two filters:

$$\mathbf{F} = \mathbf{v}\,\mathbf{h}^T$$

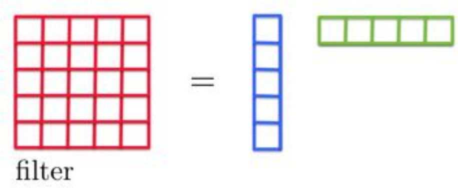
- **Homework:** Think **why** in the case of separable filters 2D convolution is the same as two 1D convolutions

[Source: R. Urtasun]

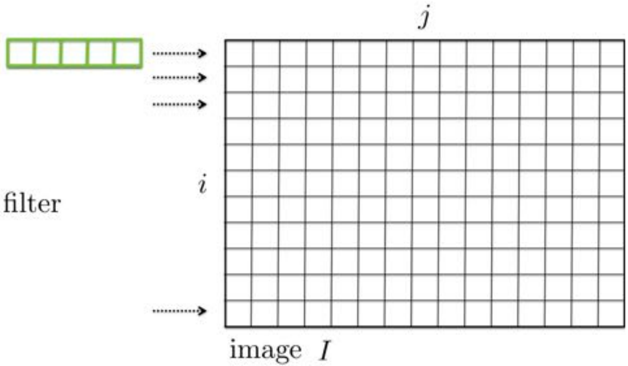
How it Works



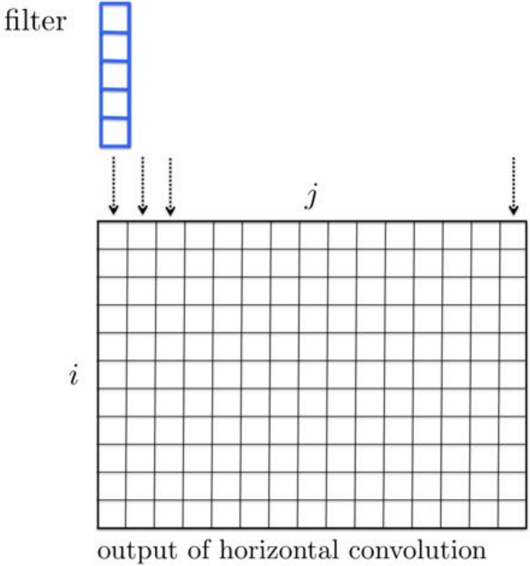
How it Works



How it Works



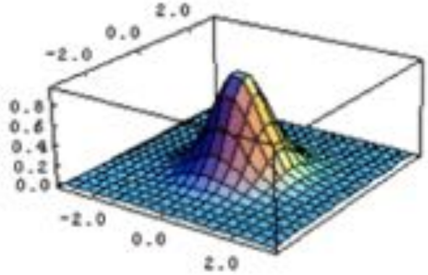
How it Works



Separable Filters: Gaussian filters

- One famous separable filter we already know:

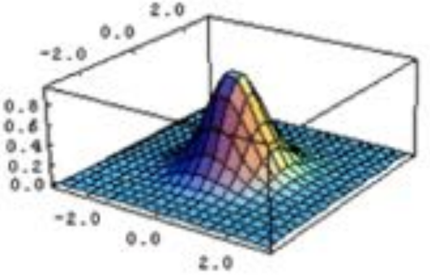
$$\text{Gaussian} \ : \ f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$



Separable Filters: Gaussian filters

- One famous separable filter we already know:

$$\text{Gaussian} \; : \; f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$
$$= \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{\sigma^2}}\right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{\sigma^2}}\right)$$



Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

[Source: R. Urtasun]

Let's play a game...

Is this separable? If yes, what's the separable version?

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \mathrm{diag}(\sigma_i)$.

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \text{diag}(\sigma_i)$.

- Matlab: $[\text{U,S,V}] = \textsc{svd}(\text{F});$

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.
- Looking at the analytic form of it.
- Look at the **singular value decomposition (SVD)**, and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \text{diag}(\sigma_i)$.

- Matlab: $[\mathrm{U,S,V}] = \textsc{svd}(\mathrm{F});$
- $\sqrt{\sigma_1}\mathbf{u}_1$ and $\sqrt{\sigma_1}\mathbf{v}_1^T$ are the vertical and horizontal filter.

[Source: R. Urtasun]

Summary – Stuff You Should Know

- **Correlation**: Slide a filter across image and compare (via dot product)
- **Convolution**: Flip the filter to the right and down and do correlation
- **Smooth** image with a Gaussian kernel: bigger σ means more blurring
- **Some** filters (like Gaussian) are **separable**: you can filter faster. First apply 1D convolution to each row, followed by another 1D conv. to each column
- Applying first a Gaussian filter with σ_1 and then another Gaussian with σ_2 is the same as applying one Gaussian filter with $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$

Functions

Python functions:

- SCIPY.NDIMAGE.CORRELATE: correlation
- SCIPY.NDIMAGE.CONVOLVE: convolution
- Many filters available: https://docs.scipy.org/doc/scipy-0.15.1/reference/ndimage.html#module-scipy.ndimage.filters

Matlab functions:

- IMFILTER: can do both correlation and convolution
- CORR2, FILTER2: correlation, NORMXCORR2 normalized correlation
- CONV2: does convolution
- FSPECIAL: creates special filters including a Gaussian

Edges

- What does blurring take away?

[Source: S. Lazebnik]

Next time:

Edge Detection